Session 302

Directory Services

David O'Rourke and Jason Townsend
Mac OS X Server Engineering

Overview

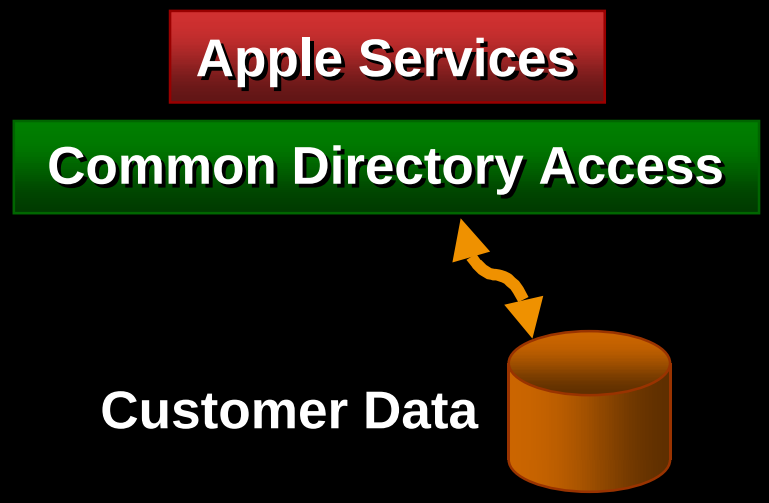
- What are Directory Services?
- How is Apple using Directory Services?
- Overview of Directory Services usage and plug-in development
- Demo
- Question and Answer

The Concept

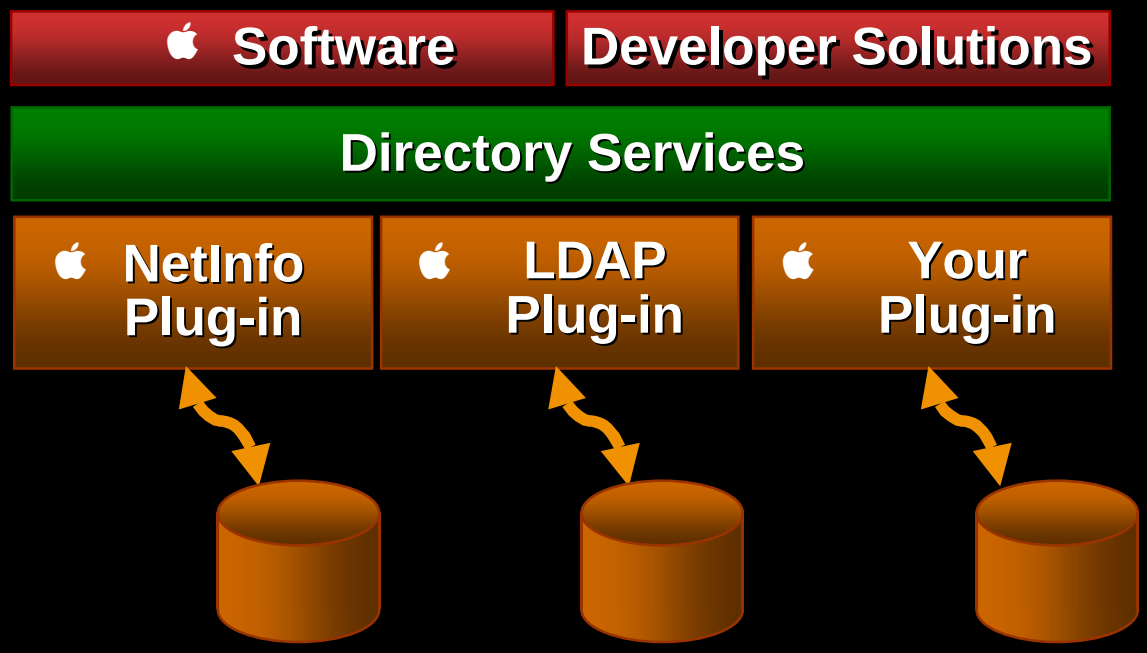
- Apple customers should be able to configure Apple software to use the customer's directory service of choice…

The Implementation

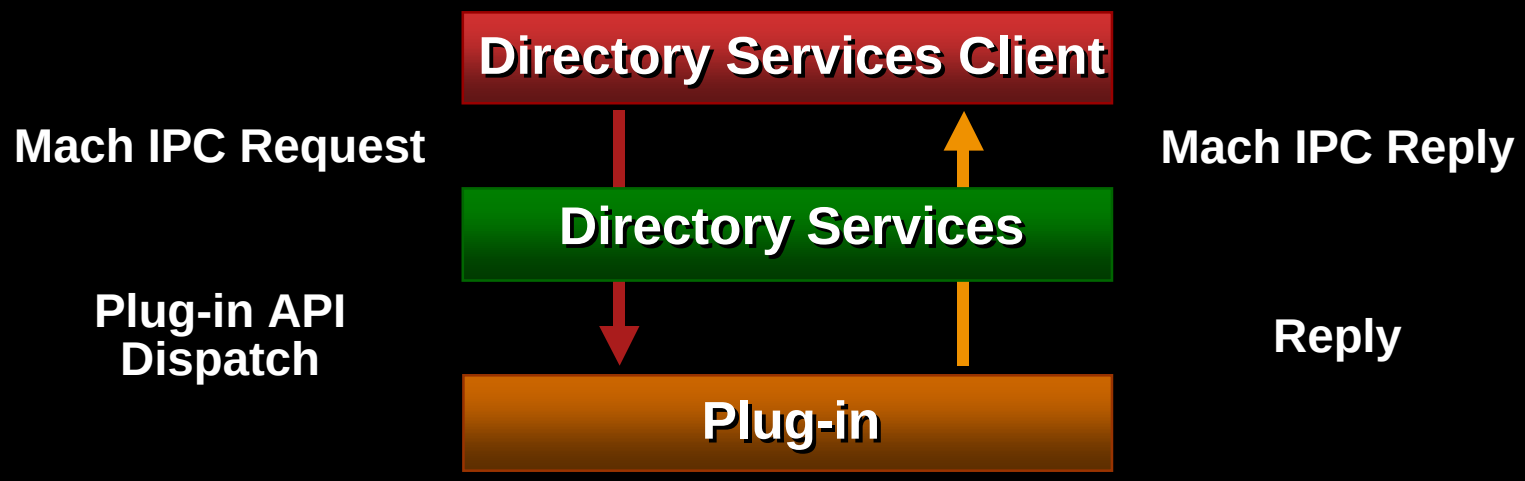
- Mac OS X is based on a common directory access architecture



The Architecture



API Runtime—Typical API Call



Status Since Last Year

- Directory Access APIs are included and functional in Mac OS X desktop and server

  /System/Library/Frameworks/DirectoryService.framework

  - API manifests itself in 5 public "C" Headers

    DirServices.h, DirServicesConst.h, DirServicesCustom.h, DirServicesTypes.h, DirServicesUtils.h

- Directory Plug-in APIs are included and functional in Mac OS X desktop and server
  - Directory plug-in SDK is available from ADR

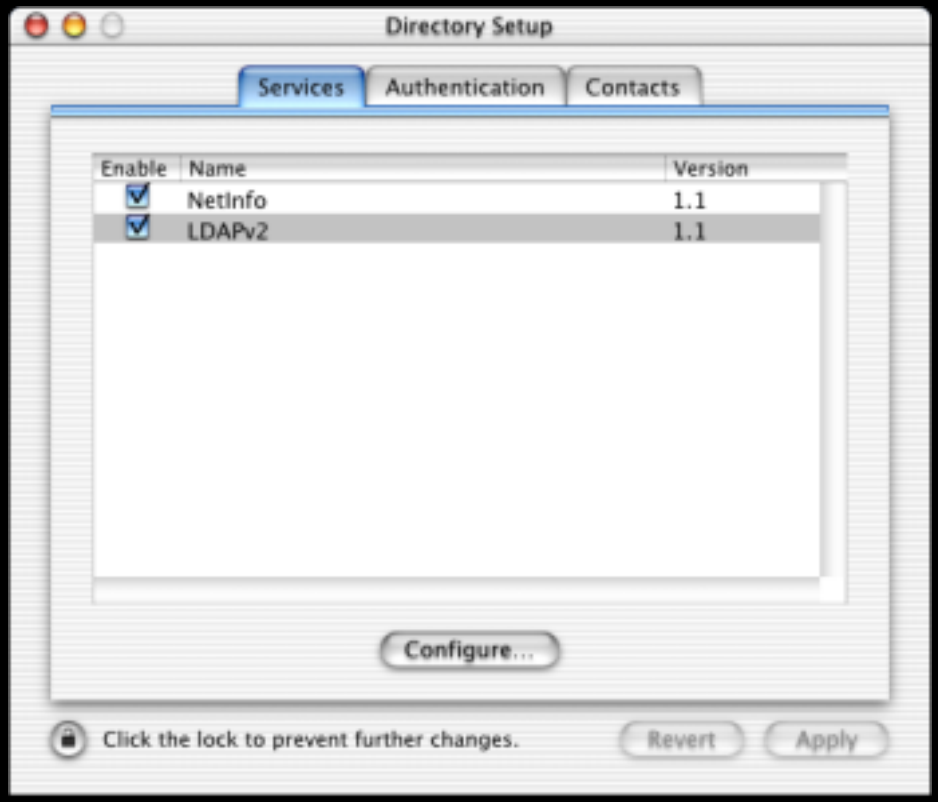
What's New Since Last Year

- Directory Setup
- New search policy
  - Both authentication and contacts search policy now available
- DSWrappers C++ sample code for easier access to the directory service APIs

Directory Setup

- Directory configuration is accomplished via Directory Setup
  - Located in/Applications/Utilities
  - Enable and disable directory access plug-ins
  - Custom plug-in configuration
    - Plug-in provides its own configuration tool, launched from Directory Setup

Screenshot


Directory Setup
Services
Authentication
Contacts
Enable
Name
Version
NetInfo
1.1
LDAPv2
1.1
Configure...
Click the lock to prevent further changes.
Revert
Apply

Access API Summary

- Full Support for Read/Write/Discovery Operations of Directory Data
- API offers: simple, complex, and custom authentication methods
- Directory APIs are standard “C” API and accessible from (Mach-O Binaries) BSD, Carbon and Cocoa on Mac OS X

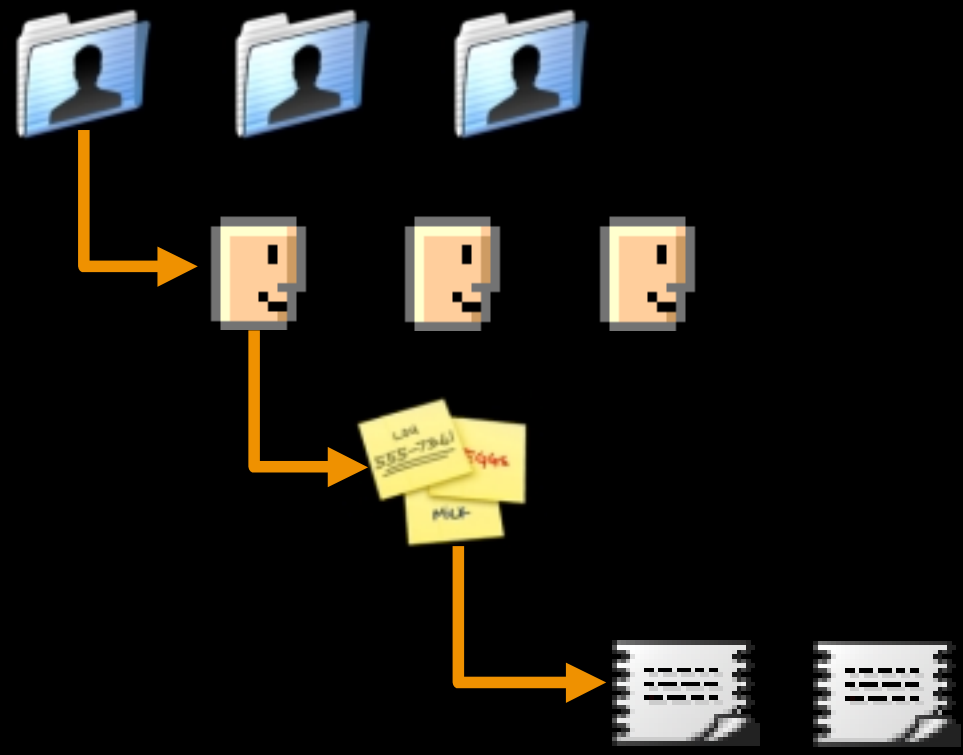
Directory Data Hierarchy

Directory Nodes

Records

Attribute Type(s)

Attribute Value(s)



Directory Nodes

- Each node is published by a plug-in
  - Plug-ins can register zero or more directory nodes
- API Calls to access directory nodes

  dsOpenDirNode(), dsGetDirNodeList(), dsFindDirNodes(), dsCloseDirNode()…
- Example path: /LDAPv2/ldap.company.com

Records

- Reside in Directory Nodes
- Consist of at least one name and exactly one type
  - Example:
    - Record Name(s): John Smith, jsmith
    - Record Type: **kDSStdRecordTypeUsers**
- API Calls to access records

  dsOpenRecord(), dsGetRecordList(), dsCreateRecord(), dsCloseRecord()…

Attribute Types

- Contained within records
- May contain zero or more values
- Example attribute types:

  kDSNAttrRecordName, kDSNAttrHomeDirectory, kDS1AttrPassword

- API calls used to access attribute types

  dsGetAttributeEntry(), dsGetRecordAttributeInfo()

Attribute Values

- Attribute values are contained within attribute types
- Attribute values contain the raw data specified by the attribute type
- API calls used to access attribute values

  dsGetAttributeValue(),
  dsGetRecordAttributeValueByIndex(),
  dsGetRecordAttributeValueByID()

Apple Adoption…

- Apple's own software
  - Administration for Mac OS X Server uses this API to configure services and obtain user/group and password information
  - Apple server software products are based on Directory Services
- 3rd-party enhancements or replacements of services can leverage Apple's directory use via these APIs

API Model—Usage…

- Apple Software products are configurable as to what directory nodes are used
  - Standard record types and their attributes are documented
  - Enables Apple and 3rd-party software to access any directory system
  - Customers configure directory access via the search policy in Directory Setup

Data Access by the OS and Applications

- Users and groups
- AutoMounting information for file system
  - NFS and AFP
- Host information
  - (Host name, IP address)
- Virtually any system configuration parameter can be stored and accessed using Directory Services

Apple Plug-in Usage and Development

- Apple's has developed 3 basic access plug-ins
  - NetInfo—Apple's own directory service
  - LDAPv2—Internet compatibility
  - Search policy plug-in
- Apple software products are being tested with these plug-ins
  - Mac OS X desktop and Mac OS X server

Beyond Apple's Basic Plug-in Access

- Apple Desktop and Server software uses this architecture to provide customer directory flexibility
- Apple is actively looking for developers interested in pursuing additional directory plug-ins

Directory Plug-in Development

Jason Townsend
Mac OS X Server Engineering

Directory Plug-in Overview

- Plug-in structure
- Plug-in testing
- Plug-in entry points
- Plug-in callbacks

Plug-in API Summary

- Plug-ins are dynamically loaded code modules
  - Directory API uses Mac OS X's CFLoad()
  - Plug-ins provide entry points for handling directory API requests
    - Validate, initialize, SetPluginState, ProcessRequest, shutdown
- Plug-ins also can call back into the Directory APIs to publish/unpublish various directories

Plug-in Binary Packaging Location

- Plug-ins are loaded from both:
  - **/System/Library/Frameworks/DirectoryService.framework/Resources/Plugins/**
  - **/Library/DirectoryServices/PlugIns/**
- Required to have a ".dsplug" extension on the bundle

Example Plug-in Bundle

- **/Library/DirectoryServices/PlugIns**
  - **<Myplugin>.dsplug -** ***(MacOS X Bundle)***
    - **<Myplugin> -** ***(Executable)***
    - **Resources -** ***(Plugin Resources)***
      - **Info-macos.plist -** ***(Required Property List)***
      - **… -** ***(Additional Plugin Resources)***

How to Make a Directory Service Plug-in

- Start with the sample Directory Plug-in code from Apple
  - Produces a single executable directory plug-in
  - Provides: stubs for all API calls, sample build process, logging examples, and API call back examples
- Add more code to access your directory service… this is where you add value

Configuring Your Plug-in

- In your plug-in's Info-macos.plist file, set the following keys:
  - **CFBundleConfigAvail**—path to an application
  - **CFBundleConfigFile**—path to a configuration file
- Directory Setup launches the specified application and/or file when you click the "Configure…" button

How to Test a Plug-in

- Edit, compile and link your plug-in source code
- Copy the Plug-in Package to one of the two Plug-in folders:
  - **/System/Library/DirectoryService.framework/Resources/Plugins/**
  - **/Library/DirectoryServices/PlugIns/**
- Stop and Restart "DirectoryService" process
- Verify your directory plug-in loaded
  - See Directory Service Log **/Library/Logs/DirectoryService.server.log**

How to Test a Plug-in

- Use Directory API calls to issue calls to your plug-in with a small test tool
  - SDK includes an example—DSTestTool
    - Simple command line driven tool
    - Easily modified to add/remove Directory API calls
  - You can develop your own API test tool to exercise your Plug-in

Plug-in Run-time Structure

- Plug-ins provide a set of standard entry points
  - **Validate()**—called once at plug-in load time
  - **Initialize()**—called once after all plug-ins are loaded
  - **SetPluginState()**—change state (enabled/disabled)
  - **ProcessRequest()**—main call for request/reply to API calls
  - **Shutdown()**— called just before Directory Service shutdown

Plug-in Callbacks

- Plug-ins can callback to Directory Services for some operations
  - **DSRegisterNode()**—used to publish a directory node
    - Register your nodes with a prefix of your plug-in's name
  - **DSUnregisterNode()**—used to unregister a registered node
  - **DSDebugLog()**—log status and debugging information
    - **/Library/Logs/DirectoryService.debug.log**

Tips—Security

- Directory Services is run as a “root” process
  - Plug-ins are considered “trusted” entities
  - Development/debugging should take place while logged on as “root”
  - Plug-in Directory is locked to root access in Mac OS X
    - System security model enforcement

Tips—Memory Usage

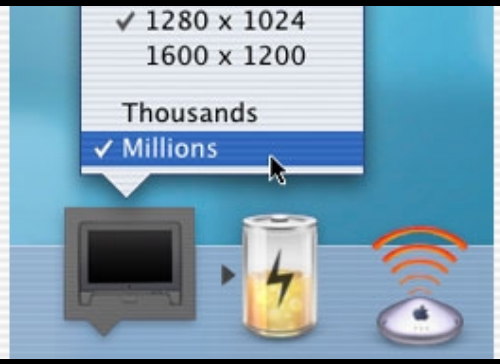
How not to leak as a Directory Services API client

- Use the appropriate calls to close references and deallocate memory
  - Use Close calls to release references
  - Use Dealloc calls to free memory
  - Use **dsReleaseContinueData()** if you don't use returned continue data in a subsequent call

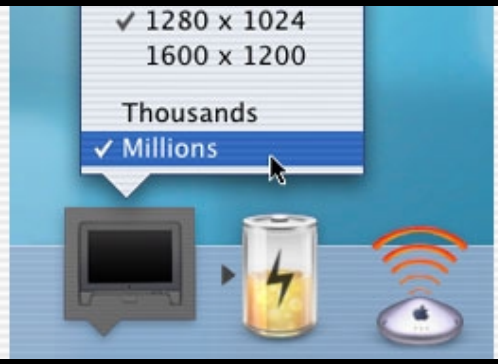
Summary of Plug-in Development

- Contact your ADR representative if you are interested in plug-in development
- Obtain both the API and Plug-in documentation
  - Review API model and being to think about how to map your directory service
- Contact Apple for access to Plug-in SDK

Demo



Directory Services Wrap-up



David O'Rourke
iServers Engineering

Directory Services and Other Apple Technologies

- Directory services complement other Apple technologies
  - Provide a common framework for higher level software to access directory data
- Network Service Location (NSL)

Opportunities. . . $$$

- Directory enabled client software
  - Enable your product to co-exist and leverage other directory enabled software
  - Provide alternatives to Apple software
  - Add value by leveraging the directory data

Opportunities…$$$

- Directory plug-ins that provides access to specific Directory Systems
  - X.500, NDS, ActiveDirectory…
  - Integrate Apple Software into existing directory infrastructure

Key Messages…

- Apple is using this architecture for its own software
  - Directory service API brings directory access to all of Apple's software products
- Apple, Developers, and Customers can leverage this API to offer maximum configuration choice
- Opportunities exist for:
  - Directory based services
  - Directory Plug-ins

Double the Opportunity…$$$

Your Product Here!

Apple's Directory Access API

Your Directory Plug-in Here!

Resources

Directory Services Plug-in and API documentation
Included with Mac OS X developer tools
/Developer/Documentation/Networking/pdf/

Thomas Weyer
Apple Worldwide Developer Relations
weyer@apple.com

Apple Developer Connection
www.apple.com/developer

Q&A

Thomas Weyer
Network & Communications Technology Manager
weyer@apple.com

http://developer.apple.com/events/urls.html

WWDC

Worldwide Developers Conference 2001